Probabilistic formulation of the classification problem

The goal in a typical classification problem is to take an input x and to assign it to one of K discrete classes C_k where $k = 1, \dots, K$ (given the dataset $D = \{(x_i, c_i)_{i=1}^n\}$

$$f: X \longrightarrow C = \{C_1, C_2, \dots, C_K\}$$

A probabilistic model do so by computing the posterior probability $P(c_i | x, D)$ = probability that the result is class c_i given a new sample x and the dataset D.

In the most common scenario the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces.
Datasets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable.

We will consider the more general case in which $X \equiv \mathbb{R}^d$ (X is a vector of real numbers): this means that the probability distributions are continuous functions.

There are two families of probabilistic models (two ways to compute $P(C_i|x, D)$ ← omitting D ($P(C_i|x)$))

- generative models: estimate $P(x|C_i)$ and then compute $P(C_i|x)$ with Bayes — in addition, they can also generate new samples
- discriminative models: estimate $P(C_i|x)$ directly

$P(x|C_i)$ → class conditional densities

generative models → Gaussian Naive Bayes

discriminative models → Logistic Regression

Probabilistic Generative Models → Gaussian Naive Bayes

- for just two classes: (concept learning)

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x)} = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

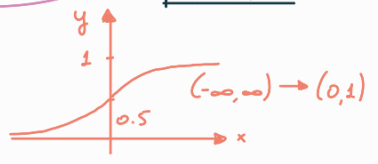
$P(C_1|x)$ (posterior); Bayes theorem

in the case of two classes we can expand the denominator with the theorem of total probabilities this way

$$a = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

$$= \frac{1}{1 + e^{-a}} = \sigma(a)$$

definition of the sigmoid function



We said that if $X \equiv \mathbb{R}^d$ (continuous vector) and $x \in X$ then the posterior probabilities are continuous functions. We assume that the probability distributions are gaussians.

$$P(x|C_i) = N(x, \mu, \Sigma)$$

μ: mean; Σ: covariance

We also assume that all classes share the same covariance matrix.
Given a vector x, the covariance matrix is a square matrix that gives the covariance between each pair of elements of the vector.

	C_1	C_2
C_1	$\sigma^2_{C_1}$	$\sigma^2(C_1, C_2)$
C_2	$\sigma^2(C_2, C_1)$	$\sigma^2_{C_2}$

Equivalent formulation for two (or more) classes →

$$a = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)} \longrightarrow a = \ln \frac{N(x, \mu_1, \Sigma)P(C_1)}{N(x, \mu_2, \Sigma)P(C_2)}$$

after a series of steps:

$$P(C_1|x) = \sigma(w^T x + w_0)$$

w and w_0 are different for other exponential distributions

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{P(C_1)}{P(C_2)}$$

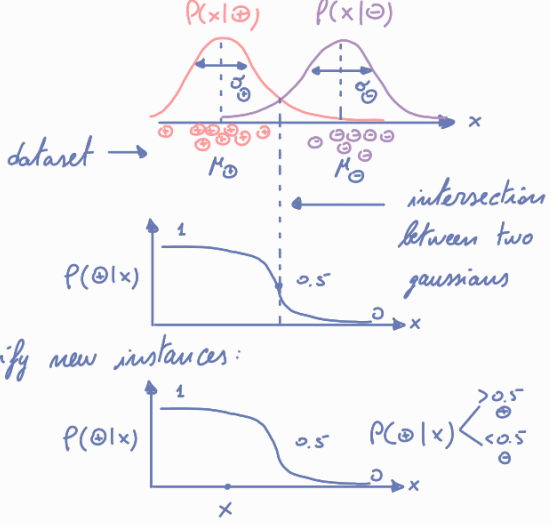
Under Gaussian assumption $P(C_1|x)$ becomes a linear expression in x (input). This is true even for other distributions that belongs to the exponential family: posterior is a linear combination of the input

this relation holds for exponential distributions →

$$a = \begin{pmatrix} — & w^T & — \end{pmatrix} \begin{pmatrix} | \\ x \\ | \end{pmatrix} + w_0$$

scalar

$f : \mathbb{R} \to \{C_1, C_2\}$



- multi-class case

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{\sum_j P(x|C_j)P(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}}$$

called softmax function

$$a = \ln P(x|C_k)P(C_k)$$

It is now easy to classify new instances:

Maximum Likelihood

- for two classes

$P(C_1) = \pi$ $\quad P(x|C_1) = N(x, \mu_1, \Sigma)$
$P(C_2) = 1-\pi$ $\quad P(x|C_2) = N(x, \mu_2, \Sigma)$

} probabilistic formulation of the concept learning problem

The parameters are π, μ_1, μ_2 (constraint with $P(C_2) = 1-\pi$)
Once we have these parameters, we can compute the posterior $P(C_i|x)$ with Bayes

Until now the dataset was always present in the formulas but omitted in order to ease the notation.

$$D = \{(x_n, t_n)_{n=1}^N\} \quad \text{with} \quad t_n = \begin{cases} 1 \text{ if } x_n \in C_1 \\ 0 \text{ if } x_n \in C_2 \end{cases}$$

$$t = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

$$P(t|\pi, \mu_1, \mu_2, \Sigma, D) = \prod_{n=1}^{N} [\pi N(x_n, \mu_1, \Sigma)]^{t_n} [(1-\pi) N(x_n, \mu_2, \Sigma)]^{(1-t_n)}$$

$P(C_1)P(x|C_1)$ $\quad$ $P(C_2)P(x|C_2)$

if $t_n = 1$ the second term is equal to 1 and only the first counts

likelihood: probability of the output given the parameters of the model and the dataset

for each sample, only one of the two gives a contribution

all the samples are independent from each other

Our problem is to find a maximum likelihood solution: estimate unknown $\pi, \mu_1, \mu_2, \Sigma$ such that the obtained solution is optimal (maximize the likelihood).

$$\underset{\pi, \mu_1, \mu_2, \Sigma}{\text{argmax}} \; P(t|\pi, \mu_1, \mu_2, \Sigma, D) \longrightarrow \underset{\pi, \mu_1, \mu_2, \Sigma}{\text{argmax}} \; \log P(t|\pi, \mu_1, \mu_2, \Sigma, D)$$

Solution (no steps):

ML solution is still an approximation (maximum given the dataset)

$$\pi \simeq \frac{N_1}{N} \quad \frac{\text{number of examples classified in } C_1}{\text{number of examples}} \quad \frac{|\{(x, C_1)\}|}{|D|}$$

$$\mu_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n x_n \quad \text{with} \quad t_n \begin{cases} 1 \text{ if } x_n \in C_1 \\ 0 \text{ if } x_n \in C_2 \end{cases}$$

mean of all the samples that belongs to class C_1

this way in μ_1 we only have samples from C_1 and in μ_2 we only have samples from C_2

$$\mu_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1-t_n) x_n$$

$$\Sigma = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2 \quad \text{with} \quad S_i = \frac{1}{N_i} \sum_{n \in C_i} (x_n - \mu_i)(x_n - \mu_i)^T \quad , \; i = 1, 2$$

- for k classes

$P(C_1) = \pi_1$ ⋮ $P(C_N) = 1 - \sum_{i=1}^{N-1} \pi_i$

$P(x|C_1) = \mathcal{N}(x, \mu_1, \Sigma)$ ⋮ $P(x|C_N) = \mathcal{N}(x, \mu_N, \Sigma)$

$D = \{(x_n, t_n)_{n=1}^N\}$ with t_n 1-of-k encoding

$t = \begin{bmatrix} 0 \dots 1 \dots 0 \end{bmatrix}$ (k columns, N rows, t_n)

one row for each sample of the dataset, one column for each class; in each row we have all zeros except a 1 in the position that corresponds to the class of the sample.

Solution (no steps):

$$\pi_i = \frac{N_i}{N}$$

$$\mu_i = \frac{1}{N_i} \sum_{n=1}^{N} t_{nk} \, x_n$$

$$\Sigma = \sum_{k=1}^{k} \frac{N_k}{N} S_k \quad \text{with} \quad S_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$

The gaussian assumption does not work well if the classes are not linearly separable

Given a generic model, suppose it has 3 parameters. The solution for the argmax is:

$\log P(x; \theta_1, \theta_2, \theta_3)$

$\frac{\partial \log P(x; \theta_1, \theta_2, \theta_3)}{\partial \theta_1} = 0 \longrightarrow \theta_1^*$

$\frac{\partial \log P(x; \theta_1, \theta_2, \theta_3)}{\partial \theta_2} = 0 \longrightarrow \theta_2^*$

$\frac{\partial \log P(x; \theta_1, \theta_2, \theta_3)}{\partial \theta_3} = 0 \longrightarrow \theta_3^*$

Probabilistic Discriminative Models → Logistic Regression

The goal in discriminative models is to directly estimate the posterior without considering the class conditional densities → we can ignore the parameters of the distribution and only use w. Thus, we will not be able to generate new samples.

- for two classes:

$D = \{(\tilde{x}_n, t_n)_{n=1}^N\}$ with $t_n \in \{0, 1\}$

$P(t|\tilde{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}$

likelihood function

product is an assumption of the independence of the data

, $y_n = P(C_1|\tilde{x}_n)$

prediction of the posterior

prediction that the model with parameters $\tilde{w}$ will give on input $\tilde{x}_n$

one particular instance of the model

compact notation

$$w^T x + w_0 = (w_0 \; w) \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$\tilde{w} = \begin{pmatrix} w_0 \\ w \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$a_k = w^T x + w_0 = \tilde{w}^T \tilde{x}$$

formulate the problem:

$$E(\tilde{w}) = \underbrace{-\ln P(t|\tilde{w})}_{\text{negative log likelihood}} = -\sum_{n=1}^{N} [t_n \ln y_n + (1 - t_n) \ln (1 - y_n)]$$

CROSS-ENTROPY (it is an error and is defined as the negative log likelihood)

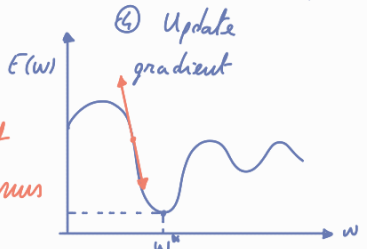
$$\text{problem} = \arg\min_{\tilde{w}} E(\tilde{w}) = \arg\max_{\tilde{w}} \ln P(t|\tilde{w})$$

We can use any solver able to minimize/maximize a function
↳ the Newton-Raphson solver is one of those

① Random value
② Compute gradient
③ Invert and follow it
④ Update

The Newton-Raphson method is an iterative optimization method.

$$\nabla E(\tilde{w}) = \sum_{n=1}^{N} (y_n - t_n)\tilde{x}_n \qquad \frac{\partial E(\tilde{w})}{\partial \tilde{w}}$$



iteratively:

$$\tilde{w} \longleftarrow \tilde{w} - H(\tilde{w})^{-1} \nabla E(\tilde{w})$$

"how much you want to move" computed in terms of the second derivative: the steeper the function, the greater the update

We move towards the minimum. It is not guaranteed to reach the global minimum; it depends on the starting point.

- multiclass logistic regression: $K \geq 2$

$f: \mathbb{R}^d \longrightarrow \{C_1, \dots, C_K\}$

$D = \{(x_n, t_n)_{n=1}^N\}$ with $t_n = (0, \dots, 0\ 1\ 0, \dots, 0)^T$ (1 in the K-th position) — 1-of-K encoding

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_N^T \end{pmatrix} \qquad T = \begin{pmatrix} t_1^T \\ \vdots \\ t_N^T \end{pmatrix}$$

1-of-K encoding in each row for the labels

$d+1$ dimensional vector, one for each input sample

We have a vector $\tilde{w}_i$ for each class: $\tilde{w}_1, \dots, \tilde{w}_K$ $\quad i \in [1, \dots, K]$

We denote with y_{nk} the prediction of model k on the sample x_n

$$y_n^T = (y_{n1}, \dots, y_{nK})^T$$ posterior prediction of $\tilde{x}_n$ for model $\tilde{w}_1, \dots, \tilde{w}_K$

$$y_{nk} = \frac{e^{\tilde{w}_k^T \tilde{x}_n}}{\sum_j e^{\tilde{w}_j^T \tilde{x}_n}}$$ one prediction for each class

$a_k = \tilde{w}_k^T \tilde{x}$; $k \in [1, \dots, K] \longrightarrow P(C_k|\tilde{x}) = \frac{e^{a_k}}{\sum_j e^{a_j}}$

$$Y(\tilde{w}_1, \dots, \tilde{w}_K)_{(N \times K)} = \begin{pmatrix} y_1^T \\ \vdots \\ y_N^T \end{pmatrix}$$

each row is a prediction vector for the respective sample: a vector of K components, one for each class. Sum of each row is = 1

Once we defined this, we can define the likelihood for the K classes case:

$$P(T|\tilde{w}_1, \dots, \tilde{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k|\tilde{x}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

with $y_{nk} = Y[n][k]$ and $t_{nk} = T[n][k]$

Cross entropy for K classes:

$$E(\tilde{w}_1, \dots, \tilde{w}_K) = -\ln P(T|\tilde{w}_1, \dots, \tilde{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

Iterative algorithm:

gradient: $\nabla_{\tilde{w}_j} E(\tilde{w}_1, \dots, \tilde{w}_K)$

hessian: $\nabla_{\tilde{w}_k} \nabla_{\tilde{w}_j} E(\tilde{w}_1, \dots, \tilde{w}_K)$

iterative step: $\tilde{w}_K \longleftarrow \tilde{w}_k - H(\tilde{w}_1, \dots, \tilde{w}_K)^{-1} \nabla E(\tilde{w}_1, \dots \tilde{w}_K)$

Generalization

① We are given a target function $f: X \to C$ and a dataset $\mathcal{D}$

② Assume a parametric prediction model $y(x, \theta)$ where θ are the parameters of the model and x is the input. y needs to be as close as possible to the target function f

$$y(x, \theta) \approx f(x)$$

③ Define an error function $E(\theta)$

④ Solve the optimization problem

$$\theta = \underset{\theta}{\operatorname{argmin}} \; E(\theta)$$

Probabilistic models

① We are given a target function $f: X \to C$ and a dataset $\mathcal{D}$

② Assume a parametric model for the posterior probability $P(C_k | \tilde{x}, \tilde{w})$

②.1 Define a posterior probability $P(C_k | \tilde{x})$

- $y_n = P(C_1 | \tilde{x}_n)$ for 2 classes
- $y_{nk} = \dfrac{e^{\tilde{w}_k^T \tilde{x}_n}}{\sum_j e^{\tilde{w}_j^T \tilde{x}_n}}$ for K classes

Posterior probability = probability of having class C given the sample and the dataset (after collecting the data)

②.2 Define the likelihood function

- $P(t | \tilde{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}$ for 2 classes
- $P(T | \tilde{w}_1, \dots, \tilde{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k | \tilde{x}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$ for K classes

③ Define an error function $E(\tilde{w})$

$$E(\tilde{w}) = - \ln P(C_k | \tilde{x}, \tilde{w})$$

④ Solve the optimization problem

$$\tilde{w}^* = \underset{\tilde{w}}{\operatorname{argmin}} \; E(\tilde{w})$$

The methods described above (Gaussian Naive Bayes + generalization) can be applied in a transformed space of the input (feature space).
Given a function $\phi: \tilde{x} \longrightarrow \phi$ (feature space), each sample $\tilde{x}_n$ can be mapped to a feature vector $\phi_n = \phi(\tilde{x}_n)$.

NB: regression = solution of a problem in which the output function is continuous;
logistic regression = classification method.